Providing Reliability as an Elastic Service in Cloud Computing

Nakharin Limrungsi, Juzi Zhao, Yu Xiang, Tian Lan, Howie Huang, Suresh Subramaniam
Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052
Email: {nakharin, juzizhao, yuxiang, tlan, howie, suresh}@gwu.edu

***Abstract*—Modern day data centers coordinate hundreds of thousands of heterogeneous tasks and aim at delivering highly reliable cloud computing services. Although offering equal reliability to all users benefits everyone at the same time, users may find such an approach either too inadequate or too expensive to fit their individual requirements, which may vary dramatically. In this paper, we propose a novel method for providing reliability as an elastic and on-demand service. Our scheme makes use of peer-to-peer checkpointing and allows user reliability levels to be jointly optimized based on an assessment of their individual requirements and total available resources in the data center. We show that the joint optimization can be efficiently solved by a distributed algorithm using dual decomposition. The solution improves resource utilization and presents an additional source of revenue to data center operators. Our validation results suggest a significant improvement of reliability over existing schemes.**

***Index Terms*—Cloud computing, data center, reliability, checkpoint, optimization.**

I. Introduction

In today's public clouds, reliability is provided as a fixed service parameter, e.g., Amazon published that its EC2 users can expect 99.95% uptime in terms of reliability, which corresponds to a once-a-week failure ratio [1]. It is up to the users to harden the tasks running within Virtual Machine (VM) instances to achieve better reliability if so desired.

Clearly, this *all-or-nothing* approach is unsatisfactory - users may find it either too inadequate or too expensive to fit their reliability requirements, which have been shown to vary dramatically [2]. Current solutions to achieve high reliability in data centers include VM replication [3], and checkpointing [4], [5], [6]. In particular, several scheduling algorithms for balancing checkpoint workload and reliability have been proposed in [7], [8], [9], with an extension in [10] by considering dynamic VM prices. Nevertheless, previous work has only investigated how to derive optimal checkpoint policies to minimize the execution time of a single task.

In this paper, we propose a novel utility-optimization approach to provide reliability as an elastic service, where flexible service-level agreements (SLAs) are made available to the users based on a joint assessment of their individual reliability requirements and total resources available in the data center. While providing reliability as a service is undoubtedly appealing to data center operators, it also comes with great technical challenges. To optimize reliability under network resource constraints, data center operators not only have to decide checkpoint scheduling, but also need to determine where to place VM checkpoints, and how to route the checkpoint traffic among peers with sufficient bandwidth. A global checkpoint scheduling (i.e., jointly determining reliability levels and checkpoint time sequences for all user) is preferred because all users share the same pool of resources. Intuitively, users with higher demands and budgets should be assigned more resources, resulting in better reliability. Their checkpoint events should also be coordinated to mitigate interference among themselves and with existing tasks. In this paper, we model different reliability requirements by user-specific utilities, which are increasing functions of reliability (i.e., service uptime). Therefore, the problem of joint reliability maximization can be formulated as an optimization, in which data center operators need to find checkpoint scheduling and make routing/placement decisions in order to maximize an aggregate utility of reliability.

This paper harnesses checkpointing technique with utility optimization to provide joint reliability maximization under resource constraints in data centers. A main feature of our approach is a peer-to-peer checkpointing mechanism, which enables VM images to be transferred and saved among neighboring peers, eliminating the need for any central storage where network congestion gets magnified across all hosts and VMs. It is demonstrated that such a distributed approach is effective to make faster checkpoints and recovery [7]. For data center operators, it also presents an additional source of revenue by exploiting under-utilized resources. For example, at any time only a few core switches are highly congested [11], which leaves adequate bandwidth among local switches for peer-to-peer traffic. Our approach can effectively convert under-utilized network resources into an on-demand reliability service, which can be purchased by users on demand.

II. Providing Reliability Through Checkpointing

A. VM Checkpointing in Data Centers

The checkpointing method is a typical fault tolerant technique in distributed systems and high-performance computing. In current data centers, Virtual Machine Monitors (VMMs) are capable of checkpointing the states of its VMs. VMMs can take local and uncoordinated checkpoints independently from each other. However, this runs the risk of cascaded rollbacks if causality is not respected. To avoid this, when a task comprises multiple VMs, taking a checkpoint of this task shall synchronously checkpoint all the VMs so that they can be rolled back to the same point of execution. Scheduling

Symbol	Meaning
n	Number of tasks, indexed by $i = 1, \ldots, n$
m_i	Number of VMs for task i
R_i	Reliability of task i
$U_i(R_i)$	Unility of task i as a function of R_i
$T_{v,i}, \eta_i$	Checkpoint interval and initial offset of task i
$T_{s,i}$	Checkpoint overhead of task i
T_r	VM rollback and recovery time
$\mathbf{X}_i$	Checkpoint routing vector of task i
$\mathscr{P}$	The set of feasible checkpoint routing vectors
$I_i(T_{v,i})$	VM delta image size as a function of $T_{v,i}$
$\mathbf{G}$	Background traffic vector
$\mathbf{C}$	Data center link capacity vector
B_i	Checkpointing bandwidth assigned for task i
λ	Node failure rate
$\mathbf{V}(t)$	Lagrangian multiplier for capacity constraints

TABLE I
MAIN NOTATION.

algorithms for taking coordinated checkpointing of a single task have been proposed in [7], [8], [9], [10]. To amortize high overhead, checkpoint intervals are often chosen to be large as long as rollback costs are acceptable.

In this paper, we assume that VMMs support a coordinated checkpointing mechanism, shown in Figure 1. For a task i with m_i VMs, checkpointing the task means synchronously checkpointing all its m_i VMs. We treat the individual VM checkpoints as a single checkpoint event with overhead $T_{s,i} = T_n + T_{b,i}$, where T_n is a constant time overhead to save local VM images and $T_{b,i}$ denotes the time to transfer the images to remote destinations. In practice, we can take T_n to be the average overhead of multiple local checkpoints [9], and $T_{b,i}$ is determined by VM image sizes to transfer and available bandwidth for checkpointing task i.

We consider a failure model that assumes independent and identical failure probabilities on all nodes (e.g., hosts). After each node failure, tasks can be recovered from the last checkpoints. All tasks using the failed node must be rolled back and restarted. We assume that failures are modeled by a Poisson process with known rate λ. Therefore, the mean time between failures is $1/\lambda$. As large-scale data centers are typically well-managed and tracked for any critical events,

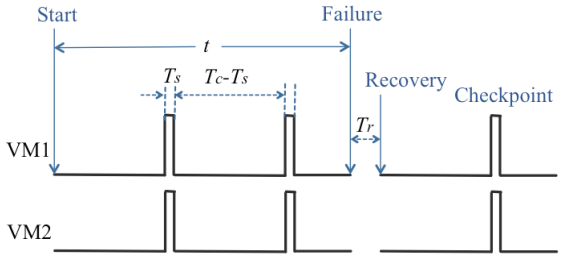

Fig. 1. Task checkpoint and recovery model. Checkpointing all VMs belonging to a task is synchronized..

the event logs can be used to provide important historical information for estimating failure rate λ.

Let $T_{v,i}$ denotes the scheduled checkpoint interval for task i, and T_r be the time overhead to roll-back to the last checkpoint, as shown in Figure 1. Through the rest of the paper, we assume that rollback time T_r is a constant. System failures can be detected by a monitoring mechanism, and failed nodes are replaced by spare ones as soon as failures are detected. Further, we consider periodic checkpointing with equal intervals $T_{v,i}$. Thus, the checkpoint time sequence of task i can be described by $T_{v,i}$ and initial time offset η_i, i.e.,

$$\eta_i,\ \eta_i + T_{v,i},\ \eta_i + 2T_{v,i},\ \ldots \tag{1}$$

which continues throughout the duration of task i.

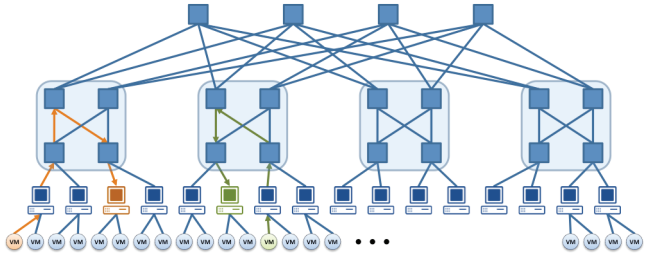

Fig. 2. Illustrations of peer-to-peer checkpointing with fat-tree topology, where traffics are distributed over the entire network and never reach top-level core switches.

B. Peer-to-Peer Checkpointing and Reliability

Mechanisms for checkpointing on central storage servers have been provided in [4], [5], [6]. Since huge amount of VM images must be transfered periodically, as the number of tasks and VMs increase in a data center, the link that connects central storage server and core switche easily becomes congested. To avoid such a bottleneck, we propose a peer-to-peer checkpointing mechanism, which enables VM images to be transferred and saved among neighboring peers. Figure 2 shows a schematic diagram of peer-to-peer checkpointing. In comparison, in a centralized checkpointing scheme where networked storage servers are connected to top-level switches, all checkpointing traffic is routed through core switches.

To characterize the benefits of peer-to-peer checkpointing, we first derive a quantification of reliability as a function of failure rate λ, and checkpoint parameters, including checkpoint overhead $T_{s,i} = T_n + T_{b,i}$, checkpoint interval $T_{v,i}$, and rollback time T_r. We define reliability by the percentage of service uptime. This yields the following Lemma 1 on the expected reliability with periodic checkpointing.

Lemma 1: If VMs of task i reside on h_i different hosts, the expected reliability of task i with periodic checkpointing interval $T_{v,i}$ is

$$\begin{aligned} R_i &= 1 - \sum_{k=1}^{\infty} \int_0^{T_{s,i}} \frac{t + kT_o + T_r + T_{v,i}}{kT_{v,i}} f_k(t)dt \\ &\quad - \sum_{k=1}^{\infty} \int_{T_{s,i}}^{T_{v,i}} \frac{t + kT_o + T_r}{kT_{v,i}} f_k(t)dt \end{aligned} \tag{2}$$

where $f_k(t) = h_i\lambda e^{-h_i\lambda[t+(k-1)T_{v,i}]}$ is the probability that a VM failure for task i occurs t seconds after the kth checkpoint interval.

Proof: Since task i uses h_i hosts, its VM failure is Possion process with rate $h_i\lambda$. Therefore, $f_k(t) = h_i\lambda e^{-h_i\lambda[t+(k-1)T_{v,i}]}$ is the p.d.f that a VM failure occurs at time $t + (k-1)T_{v,i}$.

Now if the failure occurs during $[T_{s,i}, T_{v,i}]$ of the kth checkpoint interval, the total service downtime in $kT_{v,i}$ seconds is $t + kT_o + T_r$, where the checkpointing overhead T_o is experienced in all checkpoint intervals. In contrast, if the failure occurs during $[0, T_{s,i}]$ of the k-th checkpoint interval, the total service downtime becomes $t + kT_o + T_r + T_{v,i}$, because the k-th checkpoint has not been completed yet, and task i must roll-back to the $(k-1)$-th checkpoint. Therefore, reliability is obtained as the mean percentage of service uptime as in (2). This completes the proof of Lemma 1. ∎

If we further assume that checkpoint interval $T_{v,i}$ is much smaller than the mean time between failures, i.e., $T_{v,i} \ll 1/(h_i\lambda)$, then reliability can be approximated by the following lemma:

Lemma 2: When $T_{v,i} \ll 1/(h_i\lambda)$, reliability R_i can be approximated by

$$R_i = 1 - \frac{T_o}{T_{v,i}} - h_i\lambda\left(\frac{T_{v,i}}{2} + T_r + T_{s,i}\right). \tag{3}$$

Proof: This result is straightforward by applying the approximation $e^{-h_i\lambda t} = 1$ to $f_k(t)$ on the right hand side of (2), since $t \le T_{v,i} \ll 1/(h_i\lambda)$. ∎

To illustrate limitations of the centralized checkpointing method, we consider a scenario where the link connecting central storage servers and top-level core switches is the only traffic bottleneck. This analysis provides an upper bound for the centralized checkpointing method because possible local bottlenecks are ignored. Suppose that there are n tasks with the same checkpoint interval $T_{v,i}$ and VM image size I_i. The aggregate checkpoint traffic from all tasks can not exceed the total capacity C over a checkpoint interval, i.e.,

$$\sum_{i=1}^{n} m_i I_i \le CT_{v,i} \tag{4}$$

According to (3), it implies that, for centralized checkpointing,

$$R_i \le 1 - h_i\lambda\frac{T_{v,i}}{2} \le 1 - \frac{h_i\lambda I_i}{2C}\sum_{i=1}^{n} m_i. \tag{5}$$

Reliability R_i tends to zero as the number of VMs $\sum_{i=1}^{n} m_i$ grows large. The centralized checkpointing method leads to very poor performance for large-scale data centers, where a finite bandwidth toward central storage servers is shared by a large number of VM checkpoints. This does not pose a problem for peer-to-peer checkpointing, because checkpoint traffics are distributed over local links at low-level switches, which also scales up when data center size increases. Therefore, this approach is much more scalable.

C. Problem Formulation for Optimizing Reliability

In this paper, we focus on how to determine optimal checkpoint scheduling and routing under network capacity constraints. In our model, checkpoint scheduling is decided by periodic checkpoint intervals $T_{v,i}$ and initial time offset η_i, while checkpoint routing is determined by the selection of checkpoint destination nodes and traffic routing among peers, collectively denoted by $\mathscr{P}$.

We use a utility function $U_i(\cdot)$ to model the reliability requirement of task i. A survey [2] showed that 45% of cloud users are satisfied with a 99.9% reliability guarantee (i.e., 45 minutes unplanned downtime per month), while 14% would pay at least 25% more to get a 99.99% reliability guarantee (i.e., approximately 4 minutes unplanned downtime per month), and only 6% would pay at least 50% more to go beyond 99.99%. Therefore, $U_i(R_i)$ is assumed to be an increasing function of R_i. For instance, we can choose $U_i(R_i) = -w_i\log_{10}(1-R_i)$, where w_i are user-specific weights.

For VM checkpointing, it is easy to see that task i generates periodic traffic for all

$$t \in [\eta_i + kT_{v,i} + T_o, \eta_i + kT_{v,i} + T_{s,i}],\ \forall k \in \mathbb{Z}_+. \tag{6}$$

Therefore, increasing checkpoint frequency (i.e., reducing $T_{v,i}$) generates more checkpoint traffic proportionally. Consider a data center with L links, indexed by $l = 1, \ldots, L$, each with a fixed capacity C_l. We define a checkpoint routing vector $\mathbf{X}_i$ of length L for task i by

$$\mathbf{X}_{i,l} = \begin{cases} x, & \text{if } x \text{ VM images of task } i \text{ transverse link } l, \\ 0, & \text{otherwise.} \end{cases}$$

We assume that this checkpoint routing vector $\mathbf{X}_i$ remains unchanged for the entire duration of task i.

Let B_i be the checkpoint bandwidth assigned to each VM of task i. Combining (6) and the definition of checkpoint routing vector $\mathbf{X}_i$, we can formulate a network capacity constraint as follows:

$$\mathbf{G} + \sum_{i=1}^{n} B_i\mathbf{X}_i\mathbf{1}_i(t) \le \mathbf{C},\ \forall t \tag{7}$$

where $\mathbf{C} = [C_1, \ldots, C_L]$ is a set of link capacity constraints,

and $\mathbf{1}_i(t)$ is an indicator function defined by

$$\mathbf{1}_i(t) = \mathbf{1}_{\{t \in [\eta_i + kT_{v,i} + T_o, \eta_i + kT_{v,i} + T_{s,i}], \forall k\}} \tag{8}$$

Here $\mathbf{G} = [G_1, \ldots, G_L]$ is a background traffic vector, representing the link capacities set aside for normal task traffic. An empirical measurement study in [11] shows that average traffic per VM is stable at large time-scale. Thus, we treat $\mathbf{G}$ as a time-invariant vector, where G_l denotes the aggregate task traffic on link l.

In order to create a checkpoint, only a *delta disk* [7] that contains incremental VM changes after the last checkpoint has to be saved and transferred, once the first checkpoint is done. This process considerably reduces the time needed to make the checkpoint. Therefore, we consider variable VM image sizes, as a non-decreasing function of checkpoint interval, e.g., a logarithm function $I_i(T_{v,i}) = a\log(T_{v,i}) + b$ where a, b are appropriate constants. The time to transfer VM images, $T_{b,i}$, can be computed by *delta disk* size $I_i(T_{v,i})$ and available bandwidth B_i:

$$T_{b,i} = \frac{I_i(T_{v,i})}{B_i}. \tag{9}$$

Combining (3), (7), (8) and (9), we then formulate the joint checkpoint scheduling and routing problem under network capacity constraints:

$$\text{maximize} \quad \sum_{i=1}^{n} U_i(R_i) \tag{10}$$

$$\text{subject to} \quad R_i = 1 - \frac{T_o}{T_{v,i}} - h_i\lambda\left(\frac{T_{v,i}}{2} + T_r + T_{s,i}\right) \tag{11}$$

$$\mathbf{G} + \sum_{i=1}^{n} B_i\mathbf{X}_i\mathbf{1}_i(t) \le \mathbf{C}, \ \forall t \tag{12}$$

$$T_{s,i} = T_o + \frac{I_i(T_{v,i})}{B_i} \tag{13}$$

$$\text{variables} \quad \eta_i, T_{v,i} \in \mathscr{T}, B_i, \mathbf{X}_i \in \mathscr{P} \tag{14}$$

Here we only allow users to choose $T_{v,i}$ from a finite set of checkpoint intervals, $\mathscr{T} = \{T_1, T_2, \ldots, T_z\}$. Similarly, we use $\mathscr{P}$ to denote the set of all feasible checkpoint routing vectors. Constraint (12) ensures that the required checkpoint and task traffic can be supported.

III. Solving the Joint Checkpoint Scheduling and Routing Problem

A. Our Solution Using Dual Decomposition

The problem (10) is a non-convex and combinatorial optimization and there is no computationally-efficient solution even in a centralized manner. In this paper, we leverage the technique of dual-decomposition in [12] to obtain a suboptimal solution. Among many choices of heuristic methods, the one we develop below has the advantage of allowing a distributed implementation without cooperation of different tasks.

Let M be the least common multiple of all feasible checkpoint intervals in $\mathscr{T} = \{T_1, T_2, \ldots, T_z\}$. Due to our model of periodic checkpointing, it is sufficient to consider the network capacity constraint in (12) over $[0, M]$. Let $\mathbf{V}(t)$ be a Lagrangian multiplier vector for the network capacity constraint, which is time-dependent. We derive the Lagrangian for the joint checkpoint scheduling and routing problem:

$$L = \sum_{i=1}^{n} U_i(R_i) - \int_0^M \mathbf{V}(t)^T \left[\mathbf{G} + \sum_{i=1}^{n} B_i\mathbf{X}_i\mathbf{1}_i(t) - \mathbf{C}\right] dt$$

The other two constraints (11) and (13) can be easily substituted in the Lagrangian above and are suppressed for a simple presentation.

Since M is an integer multiple of $T_{v,i}$, we have

$$\begin{aligned}
&\int_0^M \mathbf{V}(t)^T \left[\sum_{i=1}^{n} B_i\mathbf{X}_i\mathbf{1}_{\eta_i, T_{v,i}, T_{s,i}}(t)\right] dt \\
&= \sum_{i=1}^{n} \frac{MB_i}{T_{v,i}} \int_{\eta_i + T_o}^{\eta_i + T_{s,i}} \mathbf{V}(t)^T \mathbf{X}_i dt \\
&= \sum_{i=1}^{n} \frac{MI_i(T_{v,i})}{T_{v,i}} \cdot \frac{1}{T_{b,i}} \int_{\eta_i + T_o}^{\eta_i + T_{s,i}} \mathbf{V}(t)^T \mathbf{X}_i dt
\end{aligned} \tag{15}$$

where the second step uses the definition of indicator function $\mathbf{1}_i(t)$ in (8), and the last step uses $I_i(T_{v,i}) = B_i T_{b,i}$ in (9). Plugging (15) into the Lagrangian, we obtain

$$L = \sum_{i=1}^{n} \left[U_i(R_i) - \frac{MI_i(T_{v,i})}{T_{v,i}} \bar{\mathbf{V}}_i^T \mathbf{X}_i\right] - \int_0^M \mathbf{V}(t)^T \left[\mathbf{G} - \mathbf{C}\right] dt$$

where $\bar{\mathbf{V}}_i$ is an average price vector over $[\eta_i + T_o, \eta_i + T_{s,i}]$, defined by

$$\bar{\mathbf{V}}_i = \frac{1}{T_{b,i}} \int_{\eta_i + T_o}^{\eta_i + T_{s,i}} \mathbf{V}(t) dt. \tag{16}$$

Now, for given Lagrangian multiplier $\mathbf{V}(t)$, the optimization of L over checkpoint scheduling and routing is decoupled into n individual sub-problems:

$$\max_{\eta_i, T_{v,i}, B_i, \mathbf{X}_i} U_i(R_i) - \frac{MI_i(T_{v,i})}{T_{v,i}} \bar{\mathbf{V}}_i^T \mathbf{X}_i, \ \forall i \tag{17}$$

Here, the checkpoint sequence offset η_i only affects average price $\bar{\mathbf{V}}_i$, while B_i and $\mathbf{X}_i$ are fully determined by checkpoint routing/placement decisions. Thus, to solve (17) suboptimally, we can iteratively optimize it over three sets of variables: B_i and $\mathbf{X}_i$, η_i, and $T_{v,i}$, respectively. This results in the design of a heuristic and distributed algorithm for solving problem (10), if the Lagrangian multiplier $\mathbf{V}(t)$ is updated by a gradient method:

$$\mathbf{V}^{j+1}(t) = \left[\mathbf{V}^j(t) + \mu_j \left(\mathbf{G} + \sum_{i=1}^{n} B_i\mathbf{X}_i\mathbf{1}_i(t) - \mathbf{C}\right)\right]^+ \ \forall t, \tag{18}$$

where j is the interation number and μ_j is a proper stepsize.

B. Algorithm Solution for Reliability Optimization

We next present a heuristic algorithm that finds a suboptimal solution for the joint checkpoint scheduling and routing problem, leveraging the dual decomposition method presented above. The key idea is to iteratively compute the individual-user optimization problem in (17) and the price vector update

in (18). To reduce search complexity, we further break down the individual-user optimization problem in (17) into three sub-problems, over η_i, $T_{v,i}$, and $\{B_i, \mathbf{X}_i\}$, respectively. The Dijkstra algorithm is used to find the optimal routing vector $\mathbf{X}_i$ with link cost $\bar{\mathbf{V}}_i$. For a chosen tolerance λ, the proposed algorithm is summarized in Figure 3.

Initialize random interval $T_{v,i}$ and offset η_i
Intialize random routing vector $\mathbf{X}_i$ and feasible bandwidth B_i

// (a) Update price vector $\mathbf{V}(t)$*:*
$\mathbf{V}(t) \leftarrow \mathbf{V}^{s+1}(t)$ according to (18).

// (b) Solve individual-user optimization problem in (17) :
for $i = 0$ to n
  // (b.1) Search for optimal η_i*:*
  for $\eta_i \in [0, T_{v,i}]$
    Find $\eta_{i,opt}$ to minimize $\bar{\mathbf{V}}_i$ in (16)
  end for
  $\eta_i \leftarrow \eta_{i,opt}$

  // (b.2) Solve optimal B_i *and* $\mathbf{X}_i$*:*
  Treat $\bar{\mathbf{V}}_i$ as link costs
  $\mathbf{X}_i \leftarrow Dijkstra(\bar{\mathbf{V}}_i)$ for all VMs
  Assign maximum feasible B_i

  // (b.3) Search for optimal $T_{v,i}$*:*
  for $T_{v,i} \in \mathscr{T}$
    Find $T_{c,i,opt}$ to minimize $U_i(R_i) - \frac{MI_i(T_{v,i})}{T_{v,i}} \bar{\mathbf{V}}_i^T \mathbf{X}_i$,
  end for
  $T_{v,i} \leftarrow T_{c,i,opt}$
end for
Record current reliability $R_i^0 \leftarrow R_i$
Compute new R_i according to (11)
if $\sum_i |R_i - R_i^0| > \varepsilon$
  Goto (a)
end if

Fig. 3. Algorithm for joint checkpoint scheduling and routing to maximize reliability.

IV. Simulations

A. Simulation Setup

In this section, we evaluate our design for joint checkpoint scheduling and routing on a Fat-tree topology [13], which consists of a collection of edge and aggregation switches that form a complete bipartite graph (see Figure .2). While data center traffic traces are generally proprietary and unavailable, recent studies [14], [15], [16] provide us a good characterization of traffic patterns inside data centers. The major objective of our evaluation is to move a step further than analysis and obtain empirically-validated insights about the feasibility/efficiency of providing reliability as a service in practical settings.

We construct a 1024-node Fat-tree topology. The nodes are connected by 16-port high speed switches, offer a link capacity of $C_l = 1$Gbps for $l = 1, \ldots, L$. Each node represents a quad-core machine and can host up to 4 VMs. We consider a time-slotted model, so that a system snapshot is taken every $\Delta t = 10$ seconds.

To incorporate task heterogeneity, we define two types of tasks: elephant tasks that comprise $m_i = 30$ VMs and generate large peer-wise flows uniformly distributed in $[100, 200]$Mbps, and mice tasks that comprise $m_i = 5$ VMs and generate small peer-wise flows uniformly distributed in $[0, 50]$Mbps. We randomly generate $n = 300$ tasks, each being an elephant task with probability 20% and a mice task with probability 80%. Background traffic vector $\mathbf{G}$ is constructed by randomly placing all VMs in the data center and employing a shortest-path algorithm to determine their traffic routing.

Each task is associated with a utility function, given by

$$U_i(R_i) = -w_i \log_{10}(1 - R_i), \tag{19}$$

where w_i is a user-specific weight uniformly distributed in $[0, 1]$. A larger weight implies a higher reliability demand and budget. We assume that checkpoint overhead T_o is negligible and the time to take checkpoints $T_{s,i} = T_o + T_{b,i}$ is primarily determined by the time $T_{b,i}$ to transfer checkpoint images. We model checkpoint image size $I_i(T_{v,i})$ as increasing and convex functions of checkpoint interval $T_{v,i}$, i.e., $I_i(T_{v,i}) = (143 \cdot \log_{10} T_{v,i} - 254)$MB. Further, we choose rollback time $T_r = 20$ seconds, and checkpoint interval $T_{v,i}$ are selected from $\mathscr{T} = \{300, 600, 1000, 1500\}$ seconds.

B. Numerical Results

To provide benchmarks for our evaluations, we consider two heuristic algorithms with random selection of checkpoint intervals and destinations. A modified Dijkstra algorithm is employed to find maximum flow with bandwidth B_i. If B_i=0, a scheduled checkpoint event is cancelled.

1) A centralized checkpointing scheme where offset η_i is uniformly distributed in $[0, T_{v,i}]$. The link connecting central storage servers and core switches has a capacity of $C_s = 10$Gbps.
2) A peer-to-peer checkpointing scheme where offset η_i is uniformly distributed in $[0, T_{v,i}]$. All links have capacity $C_l = 1$Gbps.

Figure 4 show the p.d.f. of reliability, measured by the number-of-nines[1], for the two baseline schemes and our proposed reliability optimization algorithm. The peer-to-peer checkpointing scheme with random parameters improves reliability by roughly one order of magnitude over the centralized scheme, from 99% (i.e., two nines) to 99.9% (i.e., three nines). This is because peer-to-peer checkpointing utilizes higher bandwidth by distributing checkpoint traffic over all links. Further, our joint checkpoint scheduling and routing improves reliability by one more order of magnitude to 99.99% (i.e., four nines). Such an improvement is due to the coordination of checkpoint traffics, which becomes nearly orthogonal in temporal or spatial domain.

Figure 5 studies the impact of changing link capacity. In our proposed peer-to-peer checkpointing scheme, scaling down all link capacity to $\beta = 70\%$ expectedly reduces reliability, because it causes higher congestion in the network. However,

[1]Reliability R_i can be equivalently measured by the number-of-nines, i.e., $-\log_{10}(1 - R_i)$. For instance, three nines correspond to a reliability of 99.9%.

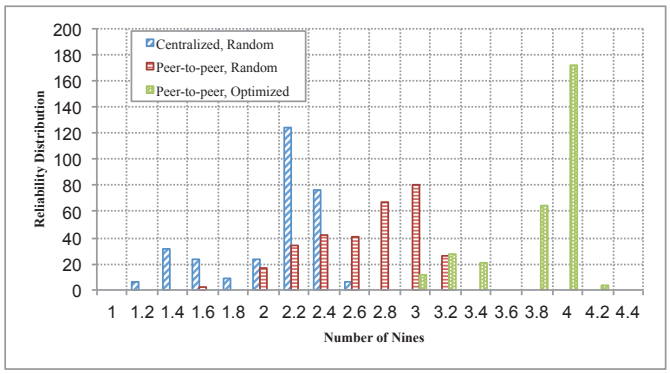

Fig. 4. Comparision of reliability on Fat-tree topology. Our proposed algorithm with peer-to-peer checkpointing shows significant reliability improvement.

the resulting performance is still better than increasing the bottleneck link capacity from $C_s = 10$Gbps to $C_s = 40$Gbps in the centralized checkpointing scheme. Peer-to-peer checkpointing and our algorithm for joint checkpoint scheduling and routing provide a cost-effective solution for achieving reliability. It mitigates the cost of deploying high capacity links in data centers.

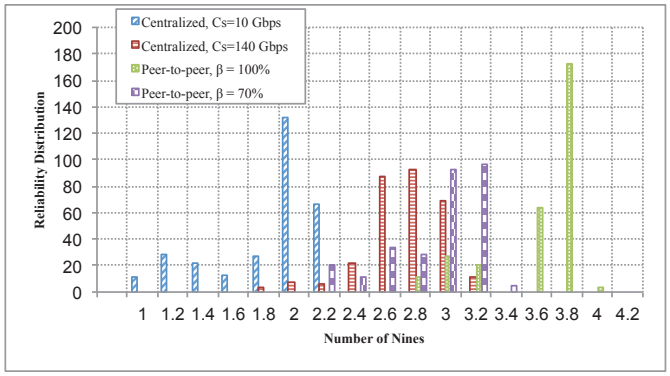

Fig. 5. Impact of changing link capacity. Our proposed algorithm with peer-to-peer checkpointing outperforms the centralized scheme even if the bottleneck link capacity is increased to $C_s = 40$Gbps.

V. Conclusion and Future Work

This paper proposes a novel approach to providing reliability as an elastic and on-demand service in cloud computing. Relying on peer-to-peer checkpointing, the problem of joint reliability maximization is formulated as an optimization, in which data center operators need to find checkpoint scheduling and make routing/placement decisions in order to maximize an aggregate utility of reliability. The resulting optimization problem, which is shown to be non-convex and combinatorial, is efficiently solved using a distributed algorithm based on dual decomposition. Numerical examples with synthesized traffic trace shows that our solution significantly improves reliability by an oder of magnitude over both random peer-to-peer and centralized checkpointing mechanisms.

In ongoing work we are looking at providing reliability as a service under dynamic job arrivals and departures, as well as for non-Poisson failure models. We are also working on reliability optimization algorithms which not only allow time-varying checkpoint scheduling (e.g., non-deterministic checkpoint intervals), but also incorporate dynamic routing/placement algorithms. We hope that the results presented in this paper provide fuel to understanding and prototyping reliability services in cloud computing.

References

[1] Amazon, " We Promise Our EC2 Cloud Will Only Crash Once A Week," *Amazon Online Technical Report*, October 2008.

[2] RackSpace, "Software as a Service Perceptions Survey", *RackSpace Technical Report*, available online at www.rackspace.com/downloads/surveys/SaaSSurvey.pdf, March 2007.

[3] VMware, "Protecting Mission-Critical Workloads with VMware Fault Tolerance," *Technical Report*, available online at www.vmware.com/files/pdf/resources/ft_virtualization_wp.pdf, February 2009.

[4] P. Ta-Shma, G. Laden, M. Ben-Yehuda, and M. Factor, "Virtual machine time travel using continuous data protection and checkpointing, *ACM SIGOPS Operating Systems Review*, vol. 42, pp. 127-134, 2008.

[5] A. Warfield, R. Ross, K. Fraser, C. Limpach, and S. Hand, "Parallax: Managing storage for a million machines, *in Proceedings of 10th Workshop on Hot Topics in Operating Systems (HotOS)*, June 2005.

[6] R. Badrinath, R. Krishnakumar, and R. Rajan, "Virtualization aware job schedulers for checkpoint-restart, *in Proceedings of 13th International Conference on Parallel and Distributed Systems (ICPADS07)*, December 2007.

[7] I. Goiri and F. Juli'a and J. Guitart and J. Torres,"Checkpoint-based Fault-tolerant Infrastructure for Virtualized Service Providers," *in Proceedings of IEEE/IFIP Network Operations and Management Symposium*, Aug 2010.

[8] M. Zhang, H. Jin, X. Shi, and S. Wu, "VirtCFT: A Transparent VM-Level Fault-Tolerant System for Virtual Clusters" *in Proceedings of Parallel and Distributed Systems (ICPADS)*, Dec 2010.

[9] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, "An Optimal Checkpoint/Restart Model for a Large Scale High Performance Computing System" *in Proceedings of Parallel and Distributed Processing (IPDPS)*, April 2008.

[10] S. Yi, D. Kondo, and A. Andrzejak, "Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud" *in Proceedings of IEEE 3rd International Conference on Cloud Computing (CLOUD)*, July 2010.

[11] J. P. Srikanth and P. Bahl, "Flyways to De-congest Data Center Networks, *IEEE 3rd International Conference on SIGCOMM Workshop on Hot Topics in Networking*, 2009.

[12] S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press, 2005.

[13] R. Mysore, A. Pamboris, and A. Vahdat, "Portland: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," *in Proceedings of ACM SIGCOMM*, 2009.

[14] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center," *in Proceedings of the 27th International Conference on Distributed Computing Systems*, ICDCS 07, pp. 59- 69, 2007.

[15] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, pp. 34-41, March 2010.

[16] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding Data Center Traffic Characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, pp. 92-99, January 2010.